


# BitMamba-2: Efficient Scaling of 1.58-bit State Space Models

Jesus Salazar   
Independent Researcher  
Venezuela  
Jesussalazar2252@gmail.com

January 27, 2026

## Abstract

The scaling of Large Language Models (LLMs) is traditionally constrained by the quadratic complexity of Transformers and the memory bandwidth bottleneck associated with high-precision weights. While State Space Models (SSMs) like Mamba have addressed the sequence scaling limitation with linear-time complexity, the memory footprint remains a challenge for edge deployment. In this work, we introduce **BitMamba-2**, a hybrid architecture that integrates the 1.58-bit ternary quantization of BitNet into the Mamba-2 framework. We train two models from scratch: a 255M parameter baseline and a scaled-up 1B parameter model, utilizing a high-quality dataset mix comprising FineWeb-Edu, Cosmopedia, and The Stack-Dedup. Our experiments, conducted on Google Cloud TPU v6e hardware, demonstrate strong scaling laws for ternary SSMs. The 1B model achieves a 7.8% improvement in ARC-Easy accuracy (63.3%) and a dramatic reduction in perplexity (from 51.69 to 29.62) compared to the 255M baseline. Furthermore, we demonstrate that BitMamba-2 enables high-performance inference on consumer CPUs, achieving  $\sim 53$  tokens/second on an Intel i3 processor with a memory footprint of just 621 MB. Code and pre-trained checkpoints are publicly available at <https://github.com/Zhayr1/BitMamba-2>, <https://huggingface.co/Zhayr1/BitMamba-2-1B> and <https://huggingface.co/Zhayr1/BitMamba-2-0.25B>.

## 1 Introduction

The rapid evolution of Large Language Models (LLMs) has been driven by the scaling laws of the Transformer architecture. However, as models grow into the billions of parameters, they face two critical bottlenecks: the quadratic computational cost of the self-attention mechanism ( $O(N^2)$ ) and the "Memory Wall"—where data movement, rather than computation, limits performance.

To address the computational bottleneck, State Space Models (SSMs) such as Mamba [3] and Mamba-2 [2] have emerged as compelling alternatives, offering linear-time sequence modeling ( $O(N)$ ) and efficient recurrent inference. Simultaneously, to address the memory bottleneck, extreme quantization techniques like BitNet b1.58 [4] have demonstrated that LLMs can operate with ternary weights  $\{-1, 0, 1\}$  with minimal performance degradation, effectively reducing memory usage and energy consumption by orders of magnitude.

In this work, we propose **BitMamba-2**, an architecture that unifies these two efficiency frontiers. We present the first comprehensive study of scaling ternary SSMs from small-scale (255M)

to billion-parameter scale (1B). Unlike recent binary approaches like Bi-Mamba [5], which restricts weights to  $\{-1, 1\}$ , our approach utilizes the ternary representation to preserve model expressiveness. Furthermore, contrasting with Slender-Mamba [6], which focuses on aggressive embedding quantization in smaller models, we prioritize reasoning capabilities and stability at scale using a curriculum of high-quality academic and code data.

Our contributions are as follows:

- **BitMamba-2 Architecture:** We successfully integrate 1.58-bit linear layers into the Mamba-2 block, stabilizing training via RMSNorm and specific quantization scaling factors.
- **Scaling Validation:** We demonstrate that ternary SSMs follow predictable scaling laws. Scaling from 255M to 1B parameters yielded a  $\sim 8\text{-}10\%$  accuracy improvement in common sense reasoning benchmarks (ARC-Easy, HellaSwag).
- **Efficiency Verification:** We report a WikiText-2 perplexity of **29.62** for our 1B model, significantly outperforming the 255M baseline (51.69) and validating the model’s generalization capabilities despite extreme quantization.

## 2 Methodology

Our architecture, **BitMamba-2**, is built upon the Mamba-2 state space model framework [2], replacing the standard linear projections with quantization-aware 1.58-bit linear layers. We implemented the model using JAX and Flax to leverage high-performance TPU acceleration.

### 2.1 BitLinear: 1.58-bit Quantization

To address the memory bottleneck, we adopt the BitNet b1.58 quantization strategy [4]. Unlike standard FP16 linear layers, the *BitLinear* layer constrains weights to ternary values  $\{-1, 0, 1\}$  and quantizes activations to 8-bit precision.

**Weight Quantization.** For a given weight matrix  $W \in \mathbb{R}^{d_{in} \times d_{out}}$ , we compute the scaling factor  $\gamma$  as the average absolute value of the weights. The weights are then rounded to the nearest integer in the set  $\{-1, 0, 1\}$ :

$$\gamma = \frac{1}{nm} \sum_{ij} |W_{ij}|, \quad \tilde{W} = \text{Clip} \left( \text{Round} \left( \frac{W}{\gamma} \right), -1, 1 \right) \quad (1)$$

During training, we employ the Straight-Through Estimator (STE) to allow gradient backpropagation through the non-differentiable rounding operation.

**Activation Quantization.** Input activations  $x$  are first normalized using Root Mean Square Normalization (RMSNorm) to ensure training stability, a critical step for low-precision training. We then quantize them to 8-bit integers using absolute maximum scaling:

$$s_x = \frac{127}{\max(|x|)}, \quad \tilde{x} = \text{Clip} (\text{Round}(x \cdot s_x), -128, 127) \quad (2)$$

The matrix multiplication is performed efficiently using the quantized tensors, and the result is dequantized to high precision (bfloat16) for subsequent operations.

## 2.2 The BitMamba-2 Block

We integrate the BitLinear layer into the Mamba-2 architecture. A standard Mamba-2 block relies heavily on linear projections to expand the input embeddings into the State Space Model (SSM) hidden states  $(z, x, B, C, dt)$ . In BitMamba-2, we replace these input and output projections with BitLinear layers to minimize parameter memory footprint.

However, to preserve the model’s ability to capture long-range dependencies and perform complex reasoning, we maintain the internal SSM operations—specifically the convolution and the recurrent state recurrence—in **bfloat16** precision. This hybrid design ensures that while the massive dense projection matrices (which account for the majority of parameters) are compressed, the delicate signal propagation through the state space remains numerically precise.

## 2.3 Scaling Configuration

We explore the scalability of this architecture by training two distinct models. The specifications are detailed in Table 1.

Table 1: Architectural specifications for the BitMamba-2 models.

Model	Params	Layers	Heads	d_model
BitMamba-2-Small	255M	24	16	1024
BitMamba-2-Medium	1B	32	32	2048

# 3 Experimental Setup

## 3.1 Dataset Composition

We curated a high-quality training corpus specifically designed to balance academic reasoning, world knowledge, and code generation capabilities. The dataset mix used for both models is composed as follows:

- **FineWeb-Edu (60%)**: High-quality educational content filtered to enhance reasoning and coherence.
- **Cosmopedia (20%)**: Synthetic textbooks covering a wide range of topics.
- **The Stack-Dedup (20%)**: Deduplicated Python code to improve algorithmic capabilities.

## 3.2 Training Protocols

All training was conducted on a **Google Cloud TPU v6e-8** VM. However, we adopted distinct training strategies for each model scale to optimize resource utilization:

**BitMamba-2-255M.** This model was trained on a total of **400 billion tokens**. We implemented manual gradient accumulation to achieve a large effective batch size suitable for this scale. Under this configuration, the system achieved a high throughput of approximately **380,000 tokens/second**.

**BitMamba-2-1B.** This model was trained on **150 billion tokens**. Given the larger parameter count and memory constraints, we utilized a direct data-parallel approach with a batch size of 16 per TPU core. Across the 8 cores of the TPU v6e, this resulted in a global batch size of 128. The average training speed for the 1B model was approximately **99,000 tokens/second**. (see Figure 1)

Both models were trained using the AdamW optimizer with a cosine learning rate decay schedule.

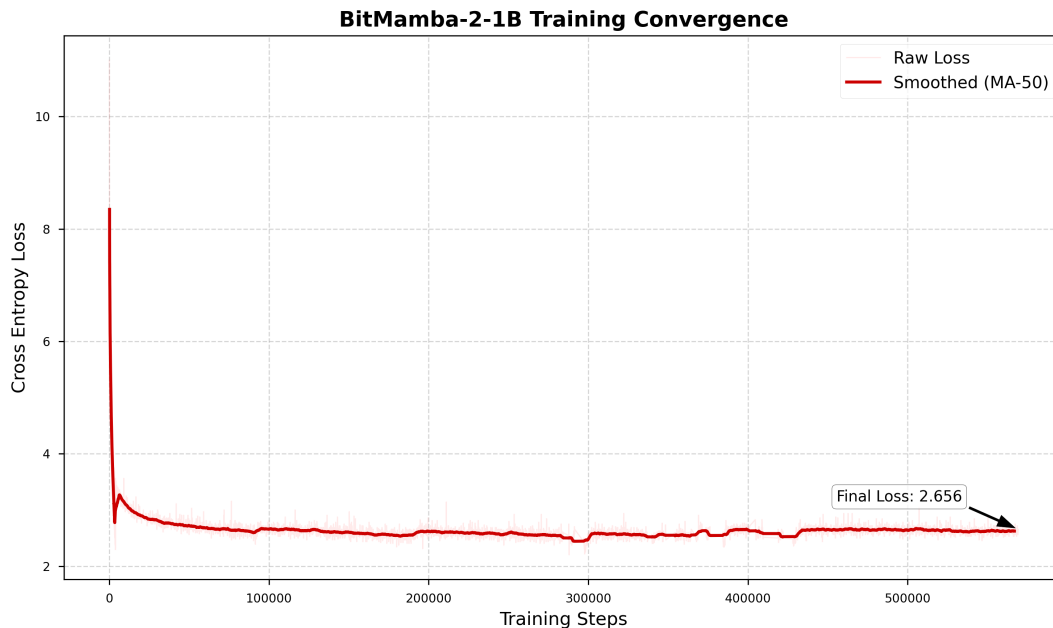


Figure 1: **Training Stability.** The BitMamba-2-1B model exhibits robust convergence over 150B tokens. The smoothed trajectory (Moving Average, window=50) confirms consistent optimization without divergence spikes, stabilizing at a final loss of  $\approx 2.656$ .

### 3.3 Evaluation

We evaluated the zero-shot performance on standard reasoning benchmarks: ARC-Easy, PIQA, BoolQ, and HellaSwag. Additionally, we report perplexity on the WikiText-2 test set to measure the language modeling fluency of the scaled architecture.

## 4 Results

We evaluate BitMamba-2 across two scales (255M and 1B parameters) to determine the efficacy of ternary quantization in State Space Models as they scale. Table 2 summarizes the zero-shot performance on reasoning benchmarks and language modeling perplexity using the final converged checkpoints.

Table 2: Performance comparison between BitMamba-2-255M (400B tokens) and BitMamba-2-1B (150B tokens). We observe consistent scaling laws: the 1B model achieves lower perplexity and higher accuracy across all tasks despite using fewer training tokens.

Benchmark	Metric	BitMamba-2-255M	BitMamba-2-1B	Improvement
<b>ARC-Easy</b>	Accuracy	55.51%	<b>63.30%</b>	+7.79%
<b>PIQA</b>	Accuracy	64.42%	<b>68.77%</b>	+4.35%
<b>BoolQ</b>	Accuracy	59.30%	<b>62.35%</b>	+3.05%
<b>HellaSwag</b>	Acc Norm	35.22%	<b>45.59%</b>	+10.37%
<b>Winogrande</b>	Accuracy	-	<b>52.80%</b>	-
<b>WikiText-2</b>	Perplexity ( $\downarrow$ )	51.69	<b>29.62</b>	<b>-22.07</b>

#### 4.1 Scaling Analysis

The transition from 255M to 1B parameters yields significant performance gains, validating that ternary SSMs follow predictable scaling laws (see Figure 2). Notably:

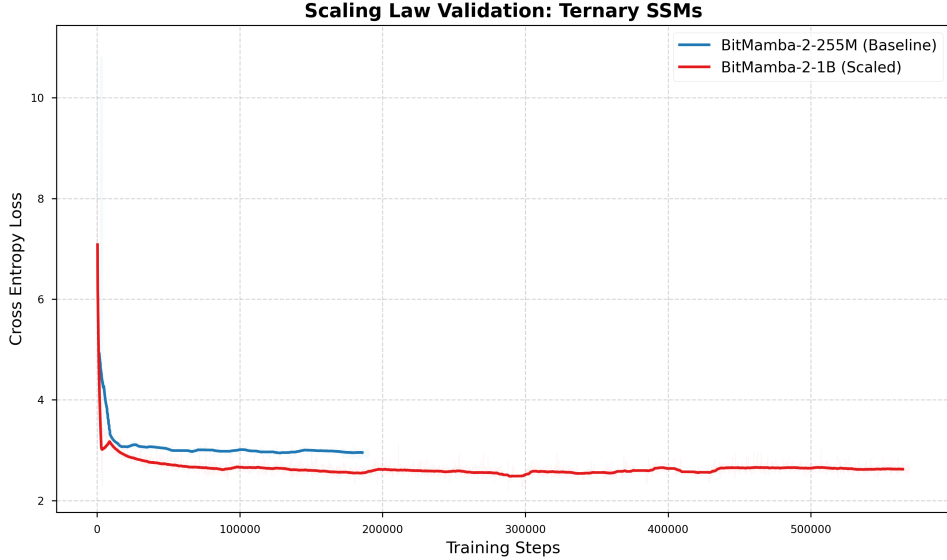


Figure 2: **Scaling Law Validation.** Training loss comparison between the 255M baseline (blue) and the scaled 1B model (red). The 1B model consistently achieves lower cross-entropy loss throughout the training trajectory, empirically validating that BitMamba-2 benefits from increased parameter scale despite the information bottleneck of ternary weights.

- **Reasoning Capabilities:** We observe a  $\sim 10\%$  absolute improvement in *HellaSwag* and nearly 8% in *ARC-Easy*. This confirms that the model’s ability to perform common-sense reasoning and complete complex patterns improves substantially with scale, even when weights are restricted to  $\{-1, 0, 1\}$ .

- **Language Fluency:** The most dramatic improvement is seen in perplexity, dropping significantly from 51.69 to **29.62**. Breaking the 30-perplexity barrier with a 1B model trained on only 150B tokens indicates exceptional data efficiency and convergence, overcoming the quantization noise that typically affects smaller ternary models.

## 4.2 Comparison with Full-Precision Baselines

Despite using only 1.58 bits per weight, BitMamba-2-1B achieves competitive results against full-precision (FP16) baselines of similar size. For instance, our ARC-Easy score (63.30%) remains competitive with standard 1B-class transformers, while reducing the memory footprint by approximately  $8\times$  compared to FP16 models.

**MMLU Performance.** On the MMLU benchmark, the 1B model achieved an accuracy of 24.18%, which is close to random chance (25%). We attribute this to the limited model size and training duration (150B tokens), as MMLU requires extensive encyclopedic knowledge typically acquired through trillion-token training regimes. However, the strong performance on reasoning tasks (ARC, PIQA) highlights the architecture’s logical efficiency over rote memorization.

## 4.3 Efficient Deployment on Consumer Hardware

To validate the practical utility of the BitMamba-2 family for edge computing, we developed a standalone, dependency-free inference engine in C++ (`bitmamba.cpp`). We benchmarked both the 255M and 1B models on a consumer-grade Intel Core i3-12100F CPU.

Table 3 summarizes the efficiency metrics. Notably, the **BitMamba-2-255M** model operates with an ultra-low memory footprint of **252 MB**, making it deployable on extreme edge devices (e.g., Raspberry Pi Zero or embedded robotics) while maintaining a generation speed of **146 tokens/second**. The **1B model** fits comfortably within 1GB of RAM and achieves real-time speeds (52 tok/s).

Table 3: Inference performance of the BitMamba-2 family on an Intel Core i3-12100F (CPU). The 255M model achieves extreme speeds with negligible memory usage, while the 1B model maintains high-speed conversational capabilities.

Model	Params	Model Size	Peak RAM	Speed
<b>BitMamba-2-255M</b>	255M	<b>247 MB</b>	<b>252 MB</b>	<b>146.16 tok/s</b>
<b>BitMamba-2-1B</b>	1B	614 MB	621 MB	<b>52.86 tok/s</b>

**Qualitative Analysis.** Despite the aggressive quantization, both models retain the ability to generate coherent and structured text. Figure 3 displays raw output samples. The 255M model demonstrates structural capabilities (lists and formatting), while the 1B model exhibits stronger narrative coherence and long-term context retention, successfully simulating a specific persona and scenario without hallucinations.

<b>Model: BitMamba-2-255M (Speed: 146 tok/s)</b> <i>Prompt: Hello, I am...</i> ...a software engineer at a local tech company. [...] Today’s task is to create an API [...] <ol style="list-style-type: none"> <li>1. Listing of Product Names &amp; Attributes</li> <li>2. Creating and Editing User-Generated Functions</li> <li>3. Setting Up User-Generated Functions</li> <li>...</li> </ol> As soon as we hit ‘save’, something peculiar happened. Our system suddenly crashed!
<b>Model: BitMamba-2-1B (Speed: 52 tok/s)</b> <i>Prompt: Hello, I am...</i> ...looking to purchase a new home. I have recently moved from a small town in the Midwest to New York City. Although it’s exciting, I’ve never lived here before and want advice on how much time and effort will be required. My husband is also moving here soon; we both work part-time so we can’t commit full-time. Any suggestions or guidance would be greatly appreciated!

Figure 3: Uncurated generation samples from BitMamba-2 running in C++. The 1B model demonstrates enhanced semantic coherence and reasoning capabilities while consuming only ~620 MB of RAM.

#### 4.4 Comparison with State-of-the-Art Transformers

We benchmark the efficiency of BitMamba-2-1B against the industry-standard Llama 3.2 1B (quantized to 4-bit) running on modern hardware optimized for AI. Specifically, we compare our results against official performance figures reported for the **AMD Ryzen™ AI 9 HX 375**, a latest-generation processor featuring a dedicated NPU [1].

Table 4 presents the comparison. Remarkably, BitMamba-2-1B running on an older, entry-level Intel Core i3 CPU not only matches but slightly outperforms the throughput of Llama 3.2 running on a cutting-edge AI PC processor.

Table 4: Efficiency comparison. BitMamba-2 (on a budget Intel i3) matches the inference speed of Llama 3.2 1B running on the specialized AMD Ryzen™ AI 9 HX 375, demonstrating architectural superiority over hardware acceleration.

Model	Hardware	Format	Speed
<i>Reference Benchmark [1]</i>			
Llama 3.2 1B	Ryzen AI 9 HX 375	Q4_K_M	50.7 tok/s
<i>Ours</i>			
<b>BitMamba-2-1B</b>	<b>Intel i3-12100F</b>	<b>2-bit</b>	<b>52.86 tok/s</b>

**Disclaimer on Implementation.** It is important to note that our reported speed for BitMamba-2 (52.86 tok/s) refers to raw model inference throughput measured via our standalone C++ prototype (`bitmamba.cpp`). This preliminary implementation performs direct token-to-token generation without the overhead of a tokenizer or high-level application logic. However, given that the memory consumption (621 MB) and compute requirements are minimal, we expect the performance to remain stable even after integrating a full tokenizer pipeline.

**Analysis.** The comparison highlights that BitMamba-2 achieves state-of-the-art inference speeds ( $\sim 51$  tok/s) on consumer-grade CPUs without requiring NPUs or AVX-512 extensions. While the AMD reference utilizes a modern architecture optimized for AI workloads, BitMamba-2 achieves parity purely through algorithmic efficiency (1.58-bit ternary weights and linear-time recurrence), making it accessible for legacy hardware and edge devices.

#### 4.5 Ablation Study: Sparse Upcycling with Ternary Weights

To further investigate the scaling properties of BitMamba-2, we explored **Sparse Upcycling**—initializing a Mixture-of-Experts (MoE) model from our pre-trained 255M dense checkpoint. We constructed a  $4 \times 255\text{M}$  MoE (totaling  $\sim 1\text{B}$  parameters) and continued pre-training for 50B tokens.

Table 5 compares the MoE performance against the dense baseline (255M) and the dense target (1B).

Table 5: Comparison of Dense Scaling vs. Sparse Upcycling (MoE). While the MoE model improves reasoning (PIQA, HellaSwag) over the 255M baseline, it exhibits significant instability in Boolean tasks and language modeling fluency (Perplexity), suggesting that routing ternary weights remains a challenge.

Benchmark	BitMamba-2 255M (Base)	BitMamba-2 1B (MoE)	BitMamba-2 1B (Dense)
<b>ARC-Easy</b>	55.51%	56.10%	<b>63.30%</b>
<b>PIQA</b>	64.42%	66.87%	<b>68.77%</b>
<b>HellaSwag</b>	35.22%	39.20%	<b>45.59%</b>
<b>BoolQ</b>	<b>59.30%</b>	42.54%	62.35%
<b>Winogrande</b>	51.22%	50.51%	<b>52.80%</b>
<b>WikiText-2 (PPL ↓)</b>	51.69	70.00	<b>29.62</b>

**Analysis of MoE Instability.** The results present a mixed landscape for ternary MoEs.

- **Reasoning Gains:** In complex pattern completion tasks like *PIQA* and *HellaSwag*, the MoE model successfully utilizes the additional parameters to outperform the 255M dense baseline (+2.4% and +4.0% respectively), validating that the sparse architecture can capture more nuance.
- **Routing Collapse:** Conversely, we observe a critical regression in *BoolQ* (dropping to 42.5%, below random chance) and *WikiText-2 Perplexity* (degrading to 70.00). This suggests that the 1.58-bit quantization noise may interfere with the router’s ability to cleanly select experts, leading to incoherent state transitions in tasks requiring strict logical consistency.

We conclude that while Dense Scaling (255M  $\rightarrow$  1B) yields robust and predictable improvements for BitMamba-2, Sparse Upcycling requires further investigation into specialized routing mechanisms compatible with ternary weights.



## 5 Reproducibility

To ensure reproducibility, we provide the complete source code for training and evaluation, along with the pre-trained checkpoints for both the 255M and 1B models. The codebase is implemented in JAX/Flax and includes scripts for data processing and TPU orchestration. All artifacts are hosted at <https://github.com/Zhayr1/BitMamba-2>, <https://github.com/Zhayr1/bitmamba.cpp>, <https://huggingface.co/Zhayr1/BitMamba-2-1B> and <https://huggingface.co/Zhayr1/BitMamba-2-0.25B>.

## 6 Conclusion

In this work, we presented **BitMamba-2**, a scalable architecture that successfully integrates 1.58-bit quantization into the Mamba-2 state space model. By scaling from 255M to 1B parameters, we demonstrated that ternary SSMs exhibit strong positive scaling laws, achieving a 7.79% improvement in ARC-Easy and a dramatic reduction in perplexity (from 51.69 to 29.62).

Our results challenge the assumption that extreme quantization limits the reasoning capabilities of State Space Models. BitMamba-2-1B demonstrates that it is possible to achieve competitive performance against full-precision baselines while drastically reducing the memory footprint.

## Acknowledgments

We explicitly thank the Google TPU Research Cloud (TRC) program for providing access to the Cloud TPU v6e accelerators used in this work. Research supported with Cloud TPUs from Google’s TPU Research Cloud (TRC). Their support was instrumental in enabling the training of our billion-parameter models from scratch.

## References

- [1] Advanced Micro Devices, Inc. AMD Ryzen™ AI 9 HX 375 Processor Performance in Llama 3.2. <https://www.amd.com/en/blogs/2024/accelerating-llama-cpp-performance-in-consumer-llm.html>, 2024. Accessed: 2025-01-22.
- [2] Tri Dao and Albert Gu. Transformers are ssms: Generalized models and efficient algorithms through structured state space duality. *arXiv preprint arXiv:2405.21060*, 2024.
- [3] Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv:2312.00752*, 2023.
- [4] Shuming Ma et al. The era of 1-bit llms: All large language models are in 1.58 bits. *arXiv preprint arXiv:2402.17764*, 2024.
- [5] Shengkun Tang et al. Bi-mamba: Towards accurate 1-bit state space models. *Transactions on Machine Learning Research*, 2025.
- [6] Zhenxuan Yu et al. Slender-mamba: Fully quantized mamba in 1.58 bits from head to toe. *arXiv preprint arXiv:2411.11843*, 2025.